Accessing NTFS partitions in Linux

ON THE DISK

Whether you are troubleshooting or just configuring for efficiency, it is a good idea to explore your options for accessing your Windows partitions from Linux. **BY OLIVER TENNERT AND JOE CASAD**

The vast quantity of Windows systems is reason enough for admins to look for ways to access an NTFS volume from Linux. And there are more urgent reasons, such as post mortem forensic investigations of Windows media after a virus infection.

Exchanging data with a Windows computer often involves reading and writing to NTFS partitions. NTFS is a modern filesystem, and thus fairly complex. To find your way around in NTFS, you'll need the right tools and a good measure of background knowledge.

Linux has had read access to NTFS for some time, but recently, a new generation of tools has evolved to offer write access, as well as support for diagnostics and troubleshooting. This month's cover story examines techniques for accessing NTFS partitions from Linux.

As you'll learn, several tools are now available for accessing NTFS. We'll start with a look at the ntfsprogs toolset, by the Linux-NTFS Project. ntfsprogs is a collection of Open Source command-line tools for mounting and managing NTFS volumes. We'll also look at Captive, a promising tool that uses Wine to access NTFS with the original Windows filesystem driver. Although Captive is free and Open Source, you'll need a Windows license to use it, since it uses the non-free Windows driver. But as the authors point out, you probably have a Windows license, or why are you using NTFS instead of the excellent Linux alternatives? The article checks out Captive in a common dual-boot scenario of a low-end system kept alive to run a legacy Windows app.

To round out our set on NTFS with Linux, Live distro guru (and Linux Magazine columnist) Klaus Knopper provides some tips for accessing an NTFS drive using a Live Linux distribution.

Some Background

Microsoft's New Technology File System (NTFS) is the standard filesystem for Windows NT, Windows XP, and Windows Server 2003. Microsoft is continuing to develop NTFS as a native filesystem for all Windows variants.

The NTFS version introduced with Windows XP is NTFS 3.1; we will be referring to this version on the following pages. The legacy FAT filesystem used in earlier versions of Windows owes its continued survival – despite its obsolete design – to mobile devices and memory cards. This said, it still plays a role in data exchange scenarios.

NTFS is far more complex than FAT, and it has a substantial feature set.

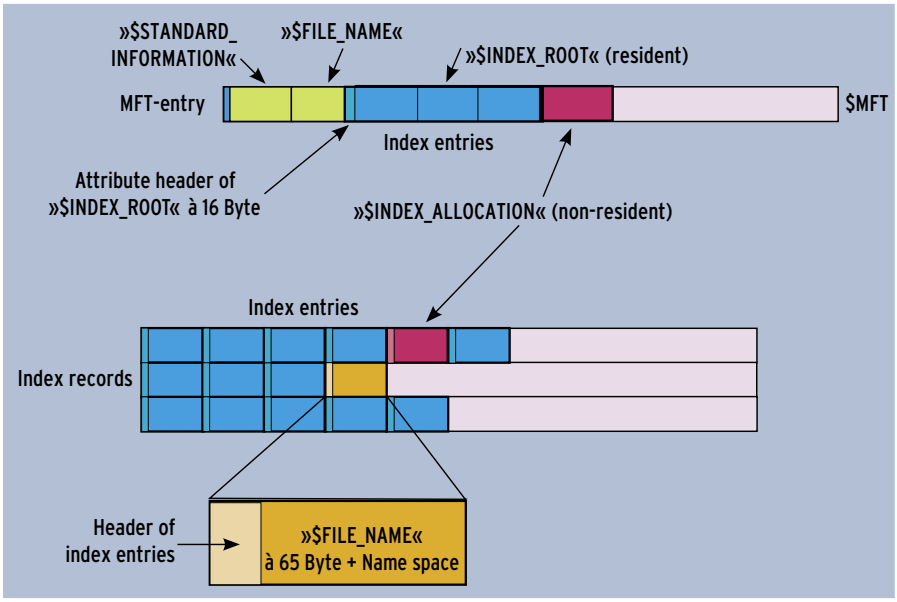

Figure 1: NTFS is far more sophisticated than the FAT filesystem used with earlier versions of Windows. In this figure, a special flag in the MFT entry header and the $FILE_NAME attribute tag this entry as a directory.

NTFS reflects Windows semantics, making it wildly different from the familiar Posix semantics used in the world of Unix. Simply mounting an NTFS filesystem doesn't make the filesystem's full feature set accessible.

On the other hand, NTFS shares a few characteristics common to modern Linux filesystems. For instance, NTFS is a journaled filesystem, and it supports metadata operations. Critical filesystem metadata included with NTFS, such as the superblock (which Microsoft refers to as the boot sector) or the Master File Table are stored redundantly to support restore operations.

In contrast to the various FAT filesystems, the NTFS filesystem has an ownership model and access control lists (ACL), whose semantics match the Windows security model far better than that of Unix.

Clustering à la Windows

When someone refers to a cluster in Windows-speak, they mean a logical block, the basic allocation unit, whose size can vary between 512 bytes and 64 kbytes depending on the volume size. Left to its own devices, Windows will set up filesystems with a maxium block size of 4 kbytes. As a 64-bit filesystem, NTFS specifies maximum file and filesystem sizes beyond today's needs.

However, as even the most recent implementations use only the lower-order 32 bits for addressing, the limit for the filesystem is about 256 terabytes, and the maximum filesize is around 16 terabytes. Just like the latest generation of Unix filesystems (XFS, JFS, and Reiser4), NTFS supports extent-based file addressing. The extent is the number of contiguous logical blocks; this is known as the cluster run in Windows-speak.

Table 1: MFT Metadata

Entry	Description
$MFT	The MFT itself
$MFTMirr	Backup of the first four MFT entries
$LogFile	Journal for logging metadata transactions
$Volume	Volume information such as the label, identifier, version, etc.
$AttrDef	Attribute information such as IDs, name, size, etc.
.	The root directory of the filesystem
$Bitmap	Allocation status for each cluster on the filesystem
$Boot	Boot sector and boot code for the filesystem
$BadClus	Clusters with bad sectors
$Secure	Information about file security and access controls
$UpCase	Uppercase version of each Unicode character
$Extend	Directory that stores files for optional, future extensions

The Mother of All Tables

The Master File Table (MFT) is the core of NTFS. The MFT is an ordered list of entries for all the files on the filesystem, which is comparable to the Inode Allocation Map on a Unix filesystem. Every file entry (file record in Microsoft-speak) on an NTFS filesystem created on Windows has a fixed size of 1 kbyte, no matter what cluster size you use. A different size could be used in theory, as the value is stored in the boot sector.

Inodes are the Unix equivalent to file records, with the exception that the inode does not store the filename, thus supporting hardlinks by simple redirection of multiple directory entries to the same inode. More complicated approaches are required for hardlinks and softlinks on NTFS.

Ironically, NTFS has a more consistent approach to the old Unix adage "everything is a file" than many Unix filesystems. The MFT lists all the metadata structures that define the filesystem as files at fixed locations. The filenames all start with a dollar sign, with one exception. Even the MFT has an entry zero with a label of *$MFT*. Table 1 lists the critical MFT metadata.

To mount an NTFS filesystem, the NTFS driver needs to know where the MFT start cluster is located on the volume. It derives this information from the first sector (sector *0* in normal computer enumeration), the 8 kbyte superblock, or boot sector, for the volume. Another MFT entry, *$Boot* (see Table 1) exists for the boot sector. Besides the filesystem metadata, the boot sector also stores the boot code required to launch Windows.

Conclusion

With this basic information in mind, read on to learn what Open Source developers have done so far to provide Linux users with access to NTFS. We hope you enjoy this month's cover story on NTFS tools for Linux. And if you're interested in the proprietary side of the NTFS puzzle, check out our Reviews section for a look at the commercial Paragon NTFS for Linux package. ■